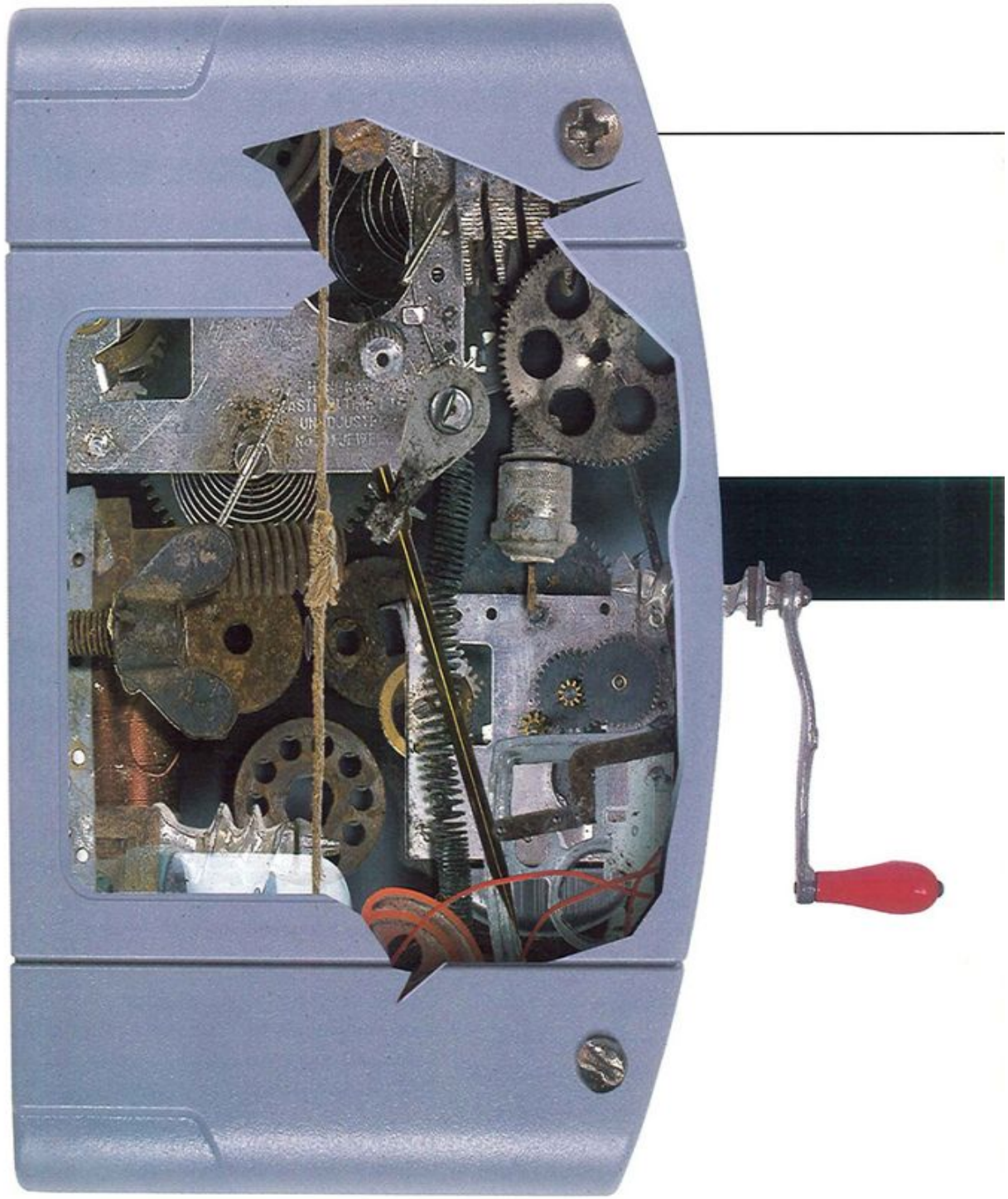
¿Cómo funcionan los juegos?

Article scanné dans le magazine Magazine 64 n°02 (Février 1998)

Sujet de l'article : Nintendo 64

Buena pregunta. Por suerte, Magazine 64 no tiene miedo a trabajo duro, y nos hemos puesto en marcha para descubrir lo que hace funcionar a los juegos de N64.



¿CÓMO FUNCIONAN LOS JUEGOS?

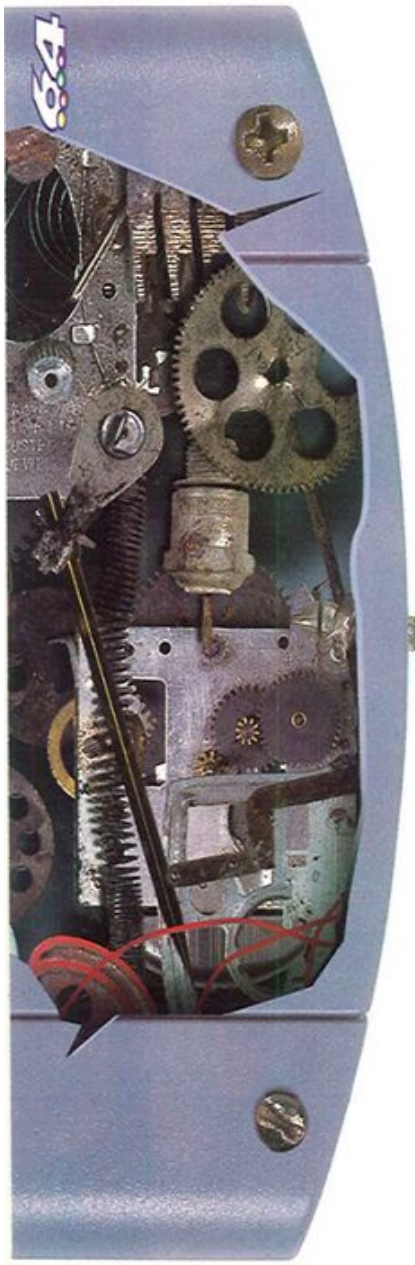
Buena pregunta. Por suerte, Magazine 64 no tiene miedo al trabajo duro, y nos hemos puesto en marcha para descubrir lo que hace funcionar a los juegos de N64.

Cuando insertas a toda prisa en la ranura de tu N64 el cartucho que acabas de comprar en la tienda de la esquina, explotan en tu pantalla las maravillas tecnológicas de la última generación de videojuegos: ¿te has preguntado alguna vez (al menos por un momento) cómo es posible que salga todo eso en tu televisor? Todos esos sonidos, los colores, las hermosas imágenes que se mueven a la velocidad del rayo, los niveles, todo el... juego. Obviamente, te imaginarás que tal despliegue de luz y sonido tiene algo que ver con los ordenadores, ¿verdad?

Durante los últimos tiempos se ha hablado mucho de las maravillas de la N64. Es algo así como una computadora Silicon Graphics reducida al tamaño de una consola. El microprocesador o CPU está compuesto por un chip de 64 bits de tipo RISC que funciona a 93,75 MHz y un coprocesador Reality, otro chip RISC de 64 bits que mueve gráficos a 62,5 MHz. Juntos pueden hacer cosas tan raras como el *anti-aliasing*, el *mip-mapping* y unos efectos de sonido muy avanzados.

Otra cosa que probablemente sabrás: todos los complicados componentes electrónicos que hay dentro de tu elegante consola, tienen algo que ver con cierto tipo de programa almacenado en el cartucho que has comprado. ¿De qué clase de programa se trata? ¿Cómo ha llegado hasta allí? ¿Cómo lo han creado? ¿Qué tipo de secretos electrónicos se han aplicado a las ideas de los diseñadores para convertirlas en un juego de verdad? ¿En qué se diferencia el programa contenido en el cartucho de los programas que utilizábamos en los Spectrum y Amstrad de los años ochenta para proyectar mensajes idiotas en pantalla? ¿Eh? ¿EH?
Sería estúpido comenzar un reportaje como éste, con preguntas como éstas, si no tuviésemos intención de responderlas. Bueno, eso de responderlas... No nos dejemos llevar por el entusiasmo. Esto es un tema complicado, y lo único que podemos prometer es que hemos hecho todo lo posible por arrojar algo de luz sobre los aspectos más interesantes del desarrollo de juegos. Estuvimos hablando con uno de los equipos que la compañía DMA Design tiene en Silicon Valley (California), y vamos a compartir con vosotros algunos de sus secretos técnicos. Sírvete un café con leche y un paquete de donuts, instálate en la silla más cómoda que tengas y disponte a descubrir todos los misterios de la N64.

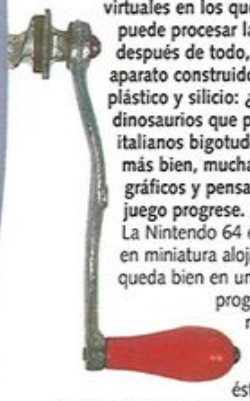
GO! GO!



Gráficos mágicos

Todas las consolas de juegos (incluyendo la N64) pasan la mayor parte del tiempo construyendo los gráficos que ves en la pantalla del televisor. ¿Qué sucede entre el momento en el que los programadores deciden que habrá un elefante verde y el instante en que el paquidermo irrumpe en tu sala de estar?

La N64 es capaz de generar auténticas maravillas visuales. Puede manejar simultáneamente un número enorme de objetos gráficos y someterlos a abundantes efectos para crear los mundos virtuales en los que jugamos. Sin embargo, aunque puede procesar las imágenes, no puede crearlas; después de todo, la consola no es más que un aparato construido cuidadosamente con metal, plástico y silicio: ¿Qué sabe la N64 sobre dinosaurios que pilotan karts o sobre personajes italianos bigotudos? Nada de nada. Alguien (o, más bien, muchas personas) tiene que dibujar los gráficos y pensar cómo variarán a medida que el juego progrese. ¿Cómo lo hacen?



La Nintendo 64 es una máquina Silicon Graphics en miniatura alojada en una carcasa de diseño que queda bien en una sala de estar, pero para programar los gráficos que decoran nuestras televisiones hace falta una de las nuevas estaciones Silicon Graphics. Y no es que éstas no queden bien con el jarrón de tu tía Tomasa, que ciertamente pega con todo, pero son un poco caras. Por lo tanto, lo más lógico es diseñar los gráficos en ordenadores Silicon Graphics. Y así lo hacen.

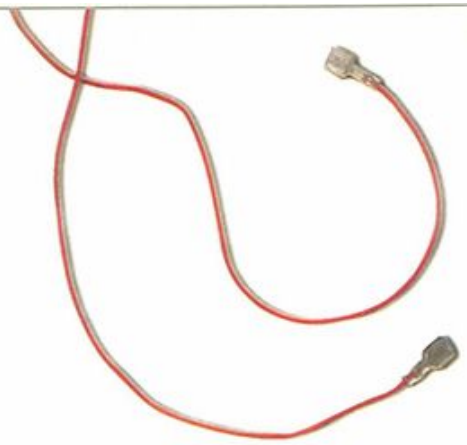
Para el modelado de polígonos 3-D, los diseñadores de la firma DMA utilizan un programa llamado Alias. Según dicen, este software es el tipo de programa de gama alta que la industria cinematográfica lleva años utilizando para crear los efectos especiales que hemos visto en películas como *Parque Jurásico* y *La Máscara*. Con este equipamiento, se diseñan y animan todos los personajes y objetos del juego.

Antes o después, los creadores quieren ver el aspecto que tienen sus geniales ideas en una N64, y es aquí donde entra otra vez en escena la alianza entre Nintendo y Silicon Graphics. Todas las estaciones de DMA están equipadas con emuladores N64 para que los gráficos puedan convertirse rápidamente a un formato que la consola comprenda, se carguen en el programa y se proyecten en pantalla tal y como aparecerían en un televisor. Si los diseñadores no están satisfechos con el resultado, pueden rehacerlo al momento, y no tienen que esperar a que aparezcan las primeras versiones del juego para estudiar si los gráficos funcionan, lo comprueban todo sobre la marcha.

No obstante, incluso en los actuales tiempos de gráficos 3-D superrealistas, el dibujo informatizado al viejo estilo tiene un lugar. La técnica tradicional de manipular imágenes píxel a píxel para crear mapas de bits es tan válida en el mundo N64 como lo era en el Spectrum; en DMA Design utilizan software más convencional, como el Adobe Photoshop (que también empleamos en *Magazine 64* para dibujar nuestras extrañas ilustraciones) y Dpaint para generar gráficos en mapas de bits, con los que se hacen las texturas y otros elementos del juego.

Por último, los desarrolladores poseen su propio software de edición de juegos, que tiene integrado un procesador de polígonos para hacer los paisajes (véase más adelante en el apartado **Construyendo un mundo**).

Cuando está todo modelado, dibujado, trazado y repasado, las imágenes se convierten a los formatos gráficos del juego, reciben nombres para que los programadores puedan manejarlos (a fin de que el programa pueda utilizar cada imagen en el momento apropiado) y, finalmente, se vinculan al código del juego. ¡Y ya está! ¿Ves qué fácil?.



Algunas de esas extrañas prestaciones gráficas

Como nos gusta pasarnos de listos, antes hemos citado algunas expresiones como mip-mapping y anti-aliasing. Esta palabrejas tan misteriosas y sugerentes son algunas de las causas por las que la N64 se ha convertido en la «magia» de final del siglo. Sin embargo, en la práctica todo es un poco más prosaico de lo que parece.

Anti-aliasing

Cuando en una imagen digital se superponen dos bloques de colores distintos y contrastados, los bordes recortados de ambos pueden aparecer en la pantalla de forma demasiado obvia. Esto tiene que

◁ *Sin anti-aliasing, las formas tienen unos bordes pixelados y recortados (izquierda).*

ver con el hecho de que cada objeto está construido a base de píxeles cuadrados, pero además hay otro factor: las frecuencias de las señales analógicas que utiliza el televisor o el monitor interfieren entre sí. Esto se denomina *aliasing*, y el resultado del *aliasing* es una imagen muy distinta a la que esperamos los usuarios modernos. O sea: que hay que hacer algo. ¿Eliminarlo tal vez? ¡Vas aprendiendo! Y para hacerlo utilizan (¿lo has adivinado ya?) el *anti-aliasing*. Los contornos de las formas se difuminan utilizando un filtro matemático cuando se dibuja la imagen en pantalla, y el resultado final es mucho más realista que una imagen a la que no se haya aplicado este tratamiento. Los bordes de los objetos 3-D se funden, haciendo parecer curvos los extremos de los polígonos y conjuntándolos de una forma mucho más natural. Así, la cara de Mario simula efectivamente una cara, y no un puzzle (como parecería en cualquier otra consola).
Los programas de dibujo por ordenador tienen esta capacidad desde hace años, pero sólo la N64 puede aplicar *anti-aliasing* en tiempo real cuando proyecta las imágenes, por lo que los mapas de bits y los polígonos que se mueven en la pantalla nunca sufren *aliasing*.

Mip-mapping

¿Qué diablos es eso? son las siglas de *multi in partem*, una expresión en latín que probablemente querrá decir algo así como «en muchas partes» (ha llovido mucho desde que estudiamos latín en 2º de BUP y, además, aprobamos copiando). La técnica del *mip-mapping* (cuyo nombre completo es nada menos que *tri-linear mip map interpolation*) sirve para cambiar el nivel de detalle de las texturas según la distancia a la que se hallan del observador, lo que hace aumentar considerablemente el realismo de las imágenes.
Como ya aprendimos en *Barrio Sésamo*, los objetos se hacen más pequeños cuando están lejos. Cambiar la escala de una imagen no representa ningún problema para un ordenador, y puede hacerse muy rápido, pero surgen problemas cuando las texturas emplean colores muy contrastados y el objeto en cuestión está muy lejos o se visualiza desde un ángulo oblicuo. En estos casos aparece lo que se conoce como (léase *muaré*), que es el mismo efecto que se observa en televisión cuando alguien lleva una camisa de rayas muy finas. Por cierto: el nombre *moiré* proviene de un efecto de diseño que se utiliza en algunos tejidos, como los de seda... ¿Cómo que no te importa, desvergonzado? En fin, sigamos...
Para evitar este problema, la N64 permite que los programadores almacenen cierto número de

△ *Gracias al mip-mapping, Goemon verás tan claras las cosas de lejos como de cerca.*

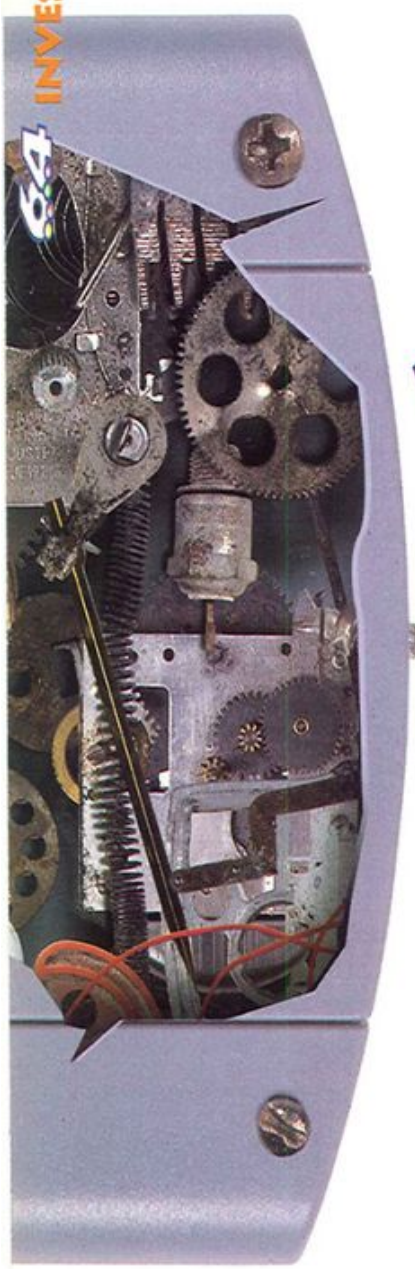
△ *Cuanto menor es la distancia a la que ves una textura, menor es su definición.*

versiones distintas de cada textura, y decide por sí misma cuál de esas versiones va a utilizar en función de la distancia y el ángulo de visión sobre el objeto. Los diseñadores de DMA Design que trabajan en Silicon Valley, por ejemplo, utilizan seis versiones de cada textura con definiciones de 32 x 32, 16 x 16, 8 x 8, 4 x 4, 2 x 2 y 1x1 píxeles respectivamente. Estas texturas están diseñadas, almacenadas y etiquetadas de forma que la N64 escoja la que quiera utilizar para proporcionar al espectador el mejor efecto visual posible.

Y hay más

Eso no es todo. La N64 decide cuántos polígonos de un objeto tiene que mostrar (lo que se denomina «control de carga»), de tal forma que puede dibujarlos muy lejanos y pequeños, en lugar de hacerlos aparecer de repente cuando se encuentra con capacidad suficiente y disponible para dibujarlos a un tamaño razonable. De la misma forma, la consola ignora los polígonos que están fuera del campo de visión de la cámara (las partes traseras de los objetos, por ejemplo) para reducir su trabajo y conseguir que la animación sea continua e ininterrumpida.

También puede manejar reflejos de luz, aplicar texturas (*mapping*) sobre objetos 3-D, trabajar con las tonalidades Gouraud para que los polígonos planos parezcan curvados (ahorrándose los increíbles cálculos matemáticos que harían falta con superficies curvas) y crear efectos de niebla. Ninguna de estas prestaciones es invento de Nintendo, ya hace mucho que se emplean en los juegos para PC. Lo excepcional es que la N64 puede hacerlo por sí misma, es decir: que todas estas características están integradas en el hardware. Los juegos para PC pueden hacer todo eso, pero sólo desde el software, y la aplicación de todos estos efectos a cada uno de los pequeños objetos que aparecen en pantalla consume ingentes recursos de la CPU, saturándola de trabajo. La velocidad de imágenes por segundo se reduce de forma alarmante cuando se apilan todos los detalles en alta resolución, hasta el punto de resultar imposible pilotar el avión/coche/lo que sea porque a dos imágenes por segundo no ves lo que está pasando. La N64 no tiene este problema gracias al coprocesador, el chip *Reality*, que se ocupa de todo el trabajo gráfico, facilitando así la tarea de los diseñadores, que sólo tienen que decir a los programadores: «quiero que esto vaya desde aquí hasta allí». Y los programadores no tienen más que decir a la computadora: «Venga, hazlo» sin pasarse noches enteras ideando un algoritmo sofisticado.



Que no pare la música

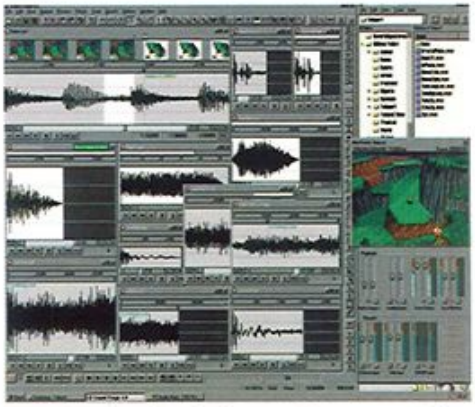
Los juegos tienen que asaltar tanto tus oídos como tus ojos, como te sucedió cuando probaste Turok. Pero sin CD, ni lector de cassette, ni nada que se le parezca... ¿de dónde salen los ruidos de la N64?

Valdría la pena detenerse un momento y tratar de recordar, en el sentido más amplio posible, qué es el sonido digital. Cada sonido viaja a través del aire en forma de ondas, es decir: fluctuaciones pequeñas y rápidas en la presión del aire. Eso ya lo sabíais... ¿verdad? Bueno, el caso es que la grabación analógica captura estas rápidas fluctuaciones de aire y las convierte en fluctuaciones de electricidad de voltaje variable. La señal eléctrica es el espectro de la onda sonora, con un voltaje que sube y baja de la misma forma que la presión del aire. Al convertirse el sonido en una pequeña descarga eléctrica, puede grabarse, manipularse o transmitirse a los altavoces. La grabación digital consiste en tomar instantáneas regulares y frecuentes de la señal y guardar el resultado de cada toma en forma de número. El resultado es una serie de pulsos eléctricos (de aquí la denominada técnica PCM o Modulación de Pulsos Codificados) que ofrecen una imagen de cómo cambia el sonido a lo largo del tiempo. Esas imágenes son las que se utilizan para reproducir el sonido (después de ciertas operaciones de filtro y manipulación, por supuesto).

El nivel de la señal en el momento de tomar la muestra se graba como un número (nota para los listillos: esto se denomina «cuantificación»). Naturalmente, cuantos más números haya donde escoger, mayor será el nivel de precisión con que se podrá grabar; y aquí es donde entra en juego la idea de «a más bits, más calidad». Si se emplea un número binario de 8 bits para grabar una señal, sólo hay 256 números posibles para cada muestra, y el resultado suena fatal. Cualquier muestra cuyo nivel se encuentre entre dos de estos valores posibles se redondeará al número más próximo y aparecerá un importante «error de cuantificación». Para reducir al mínimo este problema, se incrementa el número de bits por muestra. La N64 utiliza un sistema de calidad equivalente a la de un CD: 16 bits o, lo que es lo mismo, 65.536 valores posibles. Suena lo suficientemente parecido a la vida real como para ser indistinguible por el oído humano.

Por supuesto, si se puede grabar el sonido de forma digital también se puede sintetizar, y los ingenieros de sonido y música emplean una combinación de ambas técnicas para producir los efectos y la música de los juegos. El mundo de la música lleva años utilizando lo que se conoce como MIDI (Interfaz Digital de Instrumento Musical), que es una forma de grabar y comunicar toda una obra musical excepto el sonido, que se reproduce por medio de un sintetizador. Esto significa que se puede escribir una pieza musical y definirlo todo sobre su interpretación (la duración y el matiz de cada nota, además de algunos efectos) para después dárselo a alguien y que pueda reproducirlo en su sintetizador. Así se almacenan muchos menos datos que en una grabación de sonido, y se puede editar e interpretar de la forma que se desee con cualquier instrumento musical que el sintetizador esté preparado para emular. De esta manera, el kit que hay dentro de la N64 procesa sonidos grabados de forma digital, ondas sintetizadas que se traducen como datos de tipo MIDI. Aquí es donde se generan los espléndidos pitidos, silbidos, canciones y estruendos de los juegos.

En DMA Design utilizan el mismo tipo de instrumentos musicales que se pueden encontrar en un estudio de grabación. Un ordenador con un programa denominado Emagic Notator Logic está conectado a través de una interfaz MIDI a un sintetizador Peavey SP que, a su vez, está programado para emular a la N64. De este modo, las canciones son compuestas e interpretadas por músicos de verdad, a diferencia de lo que sucedía con los juegos clásicos, en los que eran programadores (una gente que suele carecer totalmente de oído musical) quienes componían las melodías. Los efectos de sonido se reproducen con un lector de Compact Disc que puede cambiar automáticamente 200 discos de efectos especiales (producidos por firmas como 20th Century Fox o Hanna-Barbera). Cuando han conseguido los sonidos que quieren, todo se transfiere de forma digital a una tarjeta de sonido Turtle Beach Pinnacle que está instalada en un PC Pentium equipado con el software de manipulación de audio Sound Forge. Todo el audio se modifica, se mezcla, se muestrea, se convierte y, en general, se depura hasta que se alcanzan los resultados deseados. Y después se inserta en el juego. ¡Así de simple!



△ *Aquí se está empleando el programa Forge para manipular varios sonidos distintos de un juego. Los sonidos están representados por esas líneas que parecen un electrocardiograma del director de Magazine 64 cuando se acerca el día del cierre de la revista y los colaboradores todavía no han entregado sus artículos.*

Una de las características más fascinantes de la N64 es que puede «crear» música sobre la marcha. Por ahora, sólo *Diddy Kong Racing* ha utilizado esta prestación, pero estamos seguros de que serán muchos los seguidores de esta iniciativa: la idea es conseguir bandas sonoras que interactúen con la partida que se está jugando. En *Diddy*, la música suena más rápido cuanto mayor es la velocidad. Las posibilidades son prácticamente infinitas. Alucinante.

Cómo juntarlo todo

Los gráficos ya están diseñados y la música suena a la perfección. Ahora hace falta convertirlo todo en un juego: con controles, explosiones, puntuaciones y argumento. Y entonces es cuando los programadores huyen.

Un poco de historia: el primer programa informático completo fue escrito en 1835 por Ada Byron, condesa de Lovelace (e hija de Lord Byron, el famoso poeta británico que llevaba aquellas camisas con mangas enormes). Lo escribió en tarjetas perforadas, y habría funcionado en la denominada Máquina Analítica de Charles Babbage si éste la hubiera podido construir. Desde entonces (1835, una pasada) han existido muchos ordenadores, y aún más formas de programarlos. Las cosas parecieron retroceder un poco durante los años cuarenta, cuando aparecieron los primeros ordenadores electrónicos; y la idea de poner los programas en tarjetas perforadas desapareció en breve en favor de los interruptores, conmutadores y placas de conexión. Qué aburridos. Más tarde, los programadores comenzaron con eso tan entretenido de escribir todos los ceros y unos con los que se alimentaban los ordenadores. Había nacido la «programación en código máquina». Enternecedor.

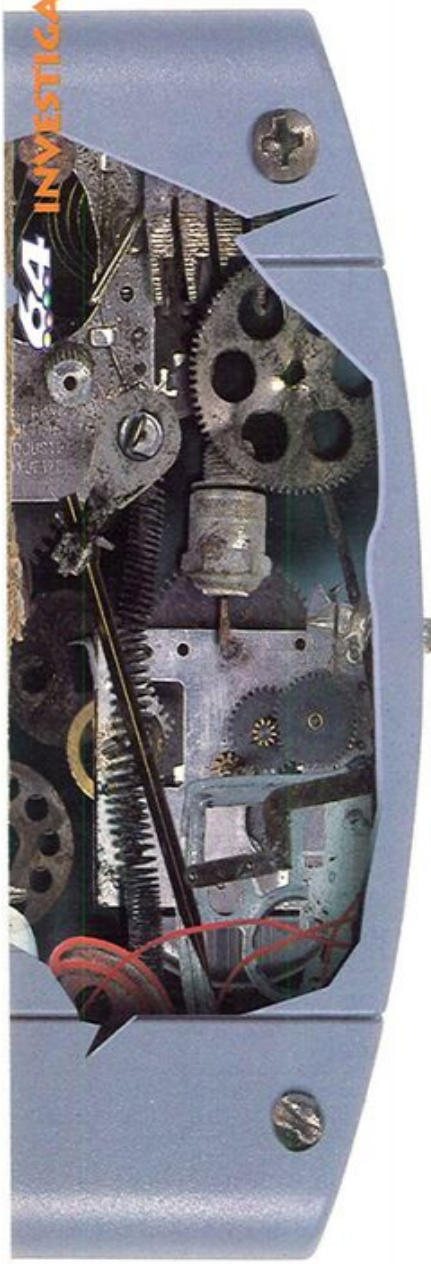
El siguiente paso fue un gran progreso en su momento, pero tampoco parece una maravilla visto desde la perspectiva actual. Se trata del «lenguaje ensamblador», en el que se emplean abreviaciones y expresiones mnemotécnicas (bastante difíciles de recordar, por cierto) para referirse a las instrucciones de los programas. Estas expresiones después se ensamblan en el «código máquina binario». Más tarde, lenguajes como FORTRAN, COBOL y ALGOL utilizaron (y, de hecho, siguen utilizando) vocablos y expresiones en inglés para construir las instrucciones. Cuando el programa está acabado, se compila en «código máquina», y el ordenador intenta ejecutarlo. Estos lenguajes son más fáciles de utilizar porque resultan más fáciles de comprender para la gente normal, y no requieren un conocimiento profundo de cómo funcionan los ordenadores. Durante las últimas décadas se han desarrollado bastante otros lenguajes de programación de alto nivel, y casi todos ellos han encontrado una aplicación en alguna especialidad informática; uno de los que han fracasado totalmente es Pascal, una especie de broma cruel que era el lenguaje enseñado a los estudiantes de ingeniería en la Universidad (esta es la razón por la que ahora los ingenieros son el hazmerreír del mundo entero).
De todos aquellos lenguajes, uno de los supervivientes es «C»: éste no sólo tiene el mérito de haber llegado a dominar la escena de la microinformática desde que lo inventaron en los años 70, también merece especial reverencia por haberlo logrado con tan desangelado nombre, con el que Dennis Ritchie (su creador) lo bautizó. Se llama «C» porque, sencillamente, su predecesor se llamaba «B». Una imaginación espléndida, Ritchie. Para poner nombre a su sucesor, se empleó otro auténtico derroche de imaginación: C++. ¿Que no te parece creativo? ¡Al menos no se llamó «D»! Un diccionario de jerga hacker disponible en Internet (http://beast.cc.emory.edu/Jargon30/jargon.html) dice lo siguiente: «C suele describirse —con una mezcla de desdén y ternura que varía en función de quien habla— como un lenguaje que combina toda la elegancia y la potencia del «ensamblador» con toda su facilidad de lectura y mantenimiento. Podríamos, naturalmente, escribir un juego en «ensamblador», pero entonces no lo podríamos publicar hasta el año 2010». «C» puede ser un infierno, pero al compilarse en «código máquina» queda limpio y eficaz, y no tiene demasiados bucles de código redundantes o inútiles que frenen la velocidad del programa.
El código se puede compilar y comprobar en los mismos emuladores N64 que se emplean para los gráficos, y si funciona se organizan fiestas y reina la alegría y el jolgorio en el departamento de desarrollo. Por si hay que depurar o actualizar el código (o incluso tomar una parte para emplearlo en otro juego), es muy importante no tener que leer todas las líneas del programa para imaginar qué hace cada instrucción. Así que, como veis en la columna de esta página, la mitad del tiempo se invierte en escribir los comentarios (las líneas que empiezan con el símbolo //) para describir lo que hace cada línea. Como decimos los redactores: «si no lo entendéis es que no estamos haciendo bien nuestro trabajo».

Un poco de Silicon Valley

¿Qué pinta tiene el código C? Aquí hay un poco que DMA ha hecho en Silicon Valley para que os hagáis una idea.

```
//////////////////////////
//////////////////////////
///////////////
// NAME : pl_UpdateDeath()
// PURPOSE : Checks if
player is dead. If so
sets up appropriate vars.

// PARAMETERS : Nothing.
//////////////////////////
//////////////////////////
///////////////
void pl_UpdateDeath( void
)
{
    if (gsPlayerInfo.Dead
== PL_DEATH_COUNTER &&
(!gPlayerIsDead))
    {

wp_StartWipe(WP_TYPE_FADE_
OFF);
    }


dead for wait amount ?
    if (gsPlayerInfo.Dead
== PL_DEATH_FADE_COUNTER
&& (!gPlayerIsDead))
    {
    // Ahh, loose a life.
    gsPlayerInfo.Lives--;
    }

    // Update Lives info.
    if
((gsPlayerInfo.Dead>
PL_DEATH_FADE_COUNTER) &&
(wa_WipeFinished()))
    {

    // Tell game to reset
energy.
    gPlayerIsDead=TRUE;

    // Set last level to
stupid val so we load a
new level.
    gLastLevel=99;

    // Player is no longer
dead.
    gsPlayerInfo.Dead=0;

    // Fade our music out
    SetSeqpFading(2, 6,
20, 0);
    SetSeqpFading(3, 6,
20, 0);

    }

    // Game Over ?
    if
(gsPlayerInfo.Lives==0)
    {

gGameAction=GA_GAME_OVER;
    gNextWave = TITLEWAVE;

gStartLevel=gsPlayerInfo.L
evel;
    }
}
```

GO! GO!



Construyendo el mundo

Así que los gráficos y el sonido ya están hechos y funcionan juntos como un juego. ¡Un momento! ¡Nos hemos olvidado de diseñar un mundo en el que tenga lugar la acción! ¡Rápido...!

Mientras en toda la oficina reina la excitación alrededor de los ordenadores carísimos que se utilizan para diseñar y programar, los niveles del juego se construyen en un pequeño rincón tranquilo decorado con muebles antiguos. Y hamacas. Aquí han enviado a paseo la alta tecnología y la calma preside el proceso creativo. Sobre los escritorios hay lápices de dureza HB y grandes hojas de papel blanco, y los escenarios del juego se diseñan SIN UTILIZAR ORDENADORES. Es asombroso que hoy día se diseñe algo, pero mucho más asombroso aún es que se diseñen cosas tan complejas como un juego. Y con un método tan rudimentario como dibujar sobre una hoja de papel..., con un lápiz. ¿Estamos de verdad en el siglo XX?

△ *SVEN es un programa diseñado especialmente para crear juegos de ordenador. Sólo funciona en equipos cuya configuración tiene un precio monstruoso.*

Pero, como ellos dicen, si un nivel no se juega sobre el papel, tampoco se jugará en una pantalla, por lo que de entrada se ahorra tiempo y energía no intentando crear niveles que no tengan ninguna posibilidad de funcionar.

Para empezar, hay que tener un concepto de juego: una idea en la que basarlo todo. Hay diseñadores trabajando en los paisajes, objetos y personajes (animales, monstruos o lo que sea). Los músicos están componiendo la banda sonora y diseñando efectos de sonido. Hay un equipo de programación que trabaja en enlazar los gráficos y el sonido dentro del concepto del juego, asegurándose de que los malos se comportan como malos y que pasa lo que tiene que pasar cuando el jugador se cae, le muerden o le aplastan la cabeza. Pero incluso entonces, cuando parece que todo está hecho, aún quedan por diseñar los niveles individuales. Y eso es lo hacen con papel y lápiz.

Bueno, al principio lo hacen así, pero cuando están seguros de que el modelo funciona sobre el papel, conjuran de nuevo a la Alta Tecnología y se ponen en manos de SVEN, que no es ningún hechicero noruego con casco de vikingo, sino un programa de edición de gráficos. Suponiendo que todo lo demás funcione más o menos bien, SVEN puede emplearse para crear paisajes, ubicar objetos y colocar los malos del juego en los sitios adecuados. Cuando se ha hecho esto, SVEN puede generar el nivel e insertarlo en el conjunto para probarlo inmediatamente en uno de esos mágicos emuladores de N64 que sirven para todo. ¿Hay que hacer algún cambio? No pasa nada, no hay que separar el código y volver a programar todo el nivel. Basta con cambiar los parámetros en SVEN y retornarlo al emulador. Este programa es un verdadero monstruo: permite cambiar casi todos los ajustes de un juego desde el confort y la familiaridad de una interfaz gráfica agradable, sin necesidad de volver a ensuciarse las manos con toda esa tontería de los entornos de programación informáticos.

¿Eso es todo?

Más o menos, sí. Crear juegos para N64 es un trabajo complejo para expertos y requiere una cantidad de dinero que asusta. En los primeros tiempos de los juegos de ordenador, cualquier chaval con un Spectrum de 30.000 pesetas podía crear un juego con posibilidades de convertirse en *best-seller*. Y muchos chicos lo hicieron. Pero a medida que pasó el tiempo, las máquinas para usuarios domésticos empezaron a ser más caras, aunque seguían estando dentro del poder adquisitivo de muchos usuarios. Las cosas se hicieron más difíciles cuando llegaron las consolas de 16 bits, aunque para crear un juego de SNES o Mega Drive sólo hacía falta un PC y un emulador.

¿Y ahora? Para comenzar a trabajar en un juego N64 se necesita, al menos, una máquina Silicon Graphics; software de emulación N64; software de modelado 3-D y un paquete de manipulación de imágenes; software de tratamiento de audio, programas de creación MIDI, sintetizadores y una buena biblioteca de discos compactos de efectos de sonido; software de diseño de escenarios de juego; una copia de «C» con un manual; y tantos diseñadores, músicos y programadores con talento como puedas encontrar. Y todo eso antes de ponerse a pensar en el juego.

La industria de los videojuegos ya no es un hobby para entusiastas de la informática. Hay quien lamenta que cada vez haya menos desarrolladores independientes de juegos que produzcan títulos innovadores en su tiempo libre, pero es el precio que tenemos que pagar por las maravillas técnicas de hoy. Con suerte, los desarrolladores independientes más afortunados pueden contar con que hay compañías (DMA Design o Rare, por ejemplo) que tienen el capital necesario para invertir en las herramientas y el personal necesarios para entrar en el desarrollo de juegos como éstos. Al menos, de ellos se puede esperar que sean personas interesadas en los juegos, y no ejecutivos que sólo se preocupan por los balances financieros de un juego hecho con prisas. Esperemos que siga funcionando así.